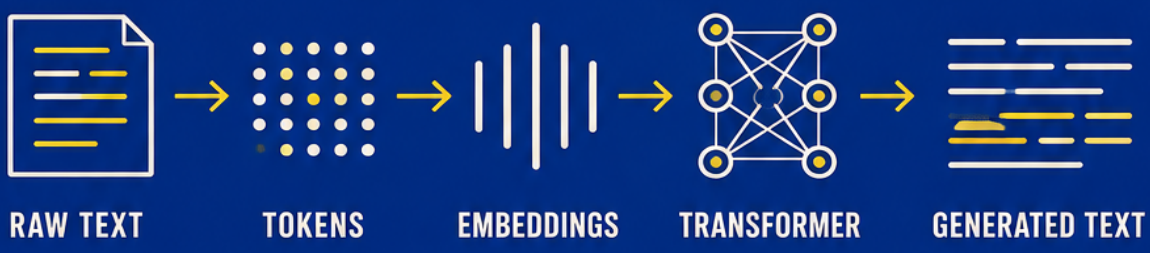
LLMs EXPLAINED

HOW LARGE LANGUAGE MODELS LEARN, THINK, AND GENERATE



CK LEE

LLMs Explained

How Large Language Models Learn, Think, and Generate

Steve Publications

This book is available at https://leanpub.com/llmsexplained

This version was published on 2026-07-17

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Steve Publications

Contents

How Large Language Models Learn, Think, and Generate

Large language models have transformed how we interact with machines, but their inner workings remain opaque to most. This book opens the black box, walking you through every stage of an LLM's life: from the raw text scraped off the internet, through tokenization and embeddings, into the transformer architecture where self-attention mechanisms weave meaning from patterns, across the grueling training process shaped by gradient descent, and finally through inference when each word is generated one step at a time. No exercises, no application checklists, no hand-waving about "artificial intelligence." Just a clear, technically precise account of what these models actually are, how they work, what they have learned, and what remains genuinely mysterious about them.

Introduction: The Black Box on Your Screen

The Question Everyone Is Asking

Type a question into a text box. A few seconds later, the screen fills with a coherent, well-structured answer in natural language. It might explain quantum mechanics, write Python code, summarize a research paper, or translate between languages you have never studied. The response feels like it came from someone who understands what you asked.

But what actually happened inside the machine?

This question has become urgent because the models that generate these answers are now everywhere: built into search engines, embedded in workplace tools, deployed as customer service agents, integrated into medical diagnostic aids, and used by millions of people daily. They are powerful, sometimes startlingly so. They are also mysterious, at least to those who did not build them. The gap between how these systems feel and how they work has created a space filled with speculation: Do they "understand"? Are they "thinking"? What is happening in the billions of parameters that make them tick?

The honest answer is both simpler and stranger than most narratives suggest. Large language models do not understand language the way humans do, and they do not think. But they are also far more than sophisticated autocomplete. They are statistical engines of extraordinary complexity, trained to predict the next token in a sequence, that have absorbed so much of human written knowledge and behavior through that single task that they can generate text which appears knowledgeable, coherent, helpful, and sometimes even creative.

This book explains how they do it. Not metaphorically, not with analogies that obscure more than they reveal, but technically, step by step, from the raw data that feeds them to the final token of their output.

What This Book Will Explain

The goal of this book is to give you a genuine, working understanding of large language models as technical systems. By the end, you should be able to answer questions like these:

- How does a model convert a sentence you type into numbers it can process?
- What is an embedding, and why does placing similar words near each other in a high-dimensional space matter?
- How does self-attention allow one word in a sentence to "know" about another word three lines earlier?
- What exactly is the model doing when it trains, and how does adjusting billions of numbers make it better at predicting text?
- Why does training on trillions of tokens give rise to abilities the model was never explicitly taught?
- How are facts "stored" inside a neural network, and what happens when the model hallucinates?
- What is instruction tuning, and how does RLHF turn a raw text predictor into something that follows directions?
- When you ask a question and get an answer, what steps occur, one token at a time, to produce that response?

The book assumes no prior knowledge of machine learning, though it will not shy away from mathematics where it clarifies rather than obscures. If you are comfortable with basic algebra and have some experience with programming, you will be able to follow along. The aim is precision without pedantry: every concept is introduced intuitively first, then made concrete.

Some topics are deliberately excluded. This book does not cover how to build applications on top of LLMs, how to prompt them for better results, or how to deploy them in production systems. Those are important subjects, but they are not what this book is about. The focus here is purely on the models themselves: their architecture, their training, their inference, and what these processes reveal about what they can and cannot do.

A First-Pass Tour: From Prompt to Response

To set the stage, let us run through a simplified version of what happens when you interact with an LLM. This tour will be brief and incomplete; every step will be unpacked in detail later.

Imagine you type the following into a chat interface: "Explain how photosynthesis works."

The first thing that happens is tokenization. Your sentence is not processed as a string of characters or even as a sequence of words. Instead, it is broken into tokens, which are fragments of text somewhere between characters and whole words. The model might split your prompt into something like: "Ex", "plain", " how", " photo", "synthesis", " works", "." Each token corresponds to a number in the model's vocabulary, which typically contains tens of thousands of entries. Your sentence is now a sequence of integers.

These integers are fed into an embedding layer, which maps each token ID to a dense vector of numbers, say 4,096 values for a modern model. These vectors are not arbitrary. Through training, the model has learned to place tokens with related meanings in similar regions of this high-dimensional space, so that "photo" and "light" end up closer together than "photo" and "banana." The embedding layer also adds positional information, because the transformer architecture treats all tokens simultaneously and would otherwise have no sense of order.

The embedded sequence then passes through the transformer layers. A modern LLM might have 32, 48, or even 96 such layers stacked on top of each other. Inside each layer, two things happen: self-attention and feed-forward computation. Self-attention is the mechanism that lets each token gather information from every other token in the sequence, weighted by relevance. When processing "works," the attention mechanism can attend strongly to "photosynthesis" because context makes them related, even though they are separated by several tokens. Feed-forward networks then transform these enriched representations through nonlinear operations, extracting higher-level patterns.

At the final layer, the model has a contextualized representation of each token in your prompt. For the last position, it produces a probability distribution over its entire vocabulary: tens of thousands of possible next tokens, each with an associated probability. The token "Photosynthesis" might have a high

probability, as might “It”, or “The”, or other plausible continuations. A decoding strategy selects one token, say “Photosynthesis.”

That token is then appended to the sequence, and the whole process repeats: the extended sequence passes through the model, a new distribution is produced, another token is selected. This loop continues until the model generates a special end-of-sequence token, or a maximum length is reached, producing your complete answer one step at a time.

This entire flow, from prompt to response, takes place with fixed parameters that were determined during training. The model does not change when it answers your question; it uses the same weights it learned from reading trillions of tokens of text scraped from the internet, books, code repositories, and other sources. Those weights encode everything the model “knows.”

How those weights come to exist is the story of training, which occupies much of this book. The core idea is deceptively simple: show the model many sequences of tokens, ask it to predict what comes next, measure how wrong it is, and adjust its parameters in a direction that reduces the error. Repeat this billions of times, and something remarkable happens. A system optimized for a single, narrow task, predicting the next token, develops the ability to answer questions, follow instructions, translate languages, write code, and perform other complex tasks it was never explicitly trained to do.

The rest of this book explains how that is possible. We will start with the fundamental prediction task, then build up through tokenization, embeddings, and the transformer architecture before diving into training mechanics, scaling laws, knowledge representation, instruction tuning, inference, and reasoning. By the end, the black box should feel considerably more like a machine you can understand.

Chapter 1: The Fundamental Idea - Predicting the Next Token

Autocomplete with Ambition

The entire capability of a large language model rests on a single task: given a sequence of tokens, predict what token comes next. That is it. No special modules for grammar, no dedicated reasoning engine, no knowledge base to query. Just next-token prediction.

This sounds like an autocomplete feature on your phone keyboard. When you type "I will see you tomor," it suggests "row." And indeed, the mechanism is related. But the scale and ambition are different in ways that matter profoundly. An LLM is not trained on a small dictionary of words with hand-crafted rules about which ones tend to follow others. It is trained on trillions of tokens drawn from an enormous diversity of sources: books, articles, forums, code repositories, scientific papers, legal documents, creative writing, technical manuals. It sees language used in every register, every domain, every style. And it is not asked merely to complete the next word; it is asked to predict the next token across all of this text, repeatedly, for hundreds of billions or trillions of examples.

Consider what the model encounters during training. It might see:

- "The capital of France is Paris"
- "def quicksort(arr):"
- "Once upon a time,"
- "H2O is composed of two hydrogen atoms and one oxygen atom."
- "The meeting has been rescheduled to 3 PM."

For each position in each sequence, the model must predict the token that follows. To do this well across such diverse material, it cannot rely on simple word associations. It needs to learn grammar so it can produce syntactically correct sentences. It needs to learn factual patterns so it can write "Paris"

after "capital of France." It needs to learn programming conventions so it can continue code. It needs to learn narrative structure so it can generate coherent stories. All of this emerges from the single pressure of minimizing prediction error.

This is the deceptively simple core insight behind LLMs: if you force a sufficiently powerful model to predict the next token across a sufficiently large and diverse corpus, it must internalize substantial structure about language and the world in order to succeed at that task. Prediction is not a trivial objective. It is, when scaled properly, an enormously demanding one.

Why Prediction Requires "Understanding"

To see why next-token prediction is harder than it first appears, let us examine some examples.

Consider the sentence: "The cat sat on the mat because it was soft." If the model has seen everything up to "because it was" and must predict the next token, what should it generate? The correct continuation depends on knowing that "it" refers to "the mat," not "the cat." A system with no sense of reference or coreference would have no basis for choosing between "soft" (describing the mat) and "tired" or "sleepy" (describing the cat). To predict correctly, the model must track what pronouns refer to across a sentence.

Now consider: "She walked into the kitchen and opened the fridge. It was empty." Again, "it" refers to the fridge. The model must maintain this reference through multiple clauses.

Or take a longer example: "John gave Mary a book. She thanked him warmly." The tokens "She" and "him" require the model to track two entities, their genders, and their roles in the previous sentence. Getting these right is not about memorizing collocations. It is about representing relationships between entities across a discourse.

These are basic linguistic phenomena, but they illustrate the point: to predict the next token accurately, a model must do work that looks a great deal like comprehension. It must:

- Parse syntax and understand sentence structure.
- Track references and resolve pronouns.

- Maintain coherence across multiple sentences.
- Apply world knowledge (fridges can be empty; cats are not typically described as "soft" in this context).
- Recognize domain conventions (code follows programming language rules; legal text has specific phrasing patterns).

None of these capabilities is explicitly taught. The model is never shown a grammar textbook or told "pronouns refer to their antecedents." It learns them implicitly because doing so improves its predictions. The training objective does not mention comprehension, but comprehension becomes instrumental to achieving the objective.

This relationship between prediction and understanding has been debated extensively. Critics argue that statistical pattern matching is not the same as genuine understanding. They are right: the model does not have beliefs, intentions, or conscious experience. But they are also missing something important. The patterns a model must learn to predict accurately are so rich and structured that the resulting system can perform tasks we intuitively associate with understanding: answering questions about texts it has read, explaining concepts, translating between languages, writing coherent arguments. Whether this counts as "understanding" depends on how you define the word. What is not in doubt is that next-token prediction, properly scaled, produces behavior that is functionally useful in contexts where understanding matters.

The Training Objective in Plain Language

Let us now state the training objective precisely, though still informally.

The model receives a sequence of tokens: t1, t2, t3, ..., tn. For each position i from 1 to n-1, it must predict the token at position i+1 based on all preceding tokens t1 through ti. This is called causal language modeling because each prediction can only depend on past information, never future information. (There are other variants of language modeling, but causal models are what power systems like GPT.)

For each position, the model produces a probability distribution over its entire vocabulary. If the vocabulary has 50,000 tokens, the model outputs 50,000 numbers that sum to 1. Each number represents the model's estimated probability that a particular token is next.

The actual next token is known during training, because it comes from real text. The model's prediction is compared to reality using a loss function called cross-entropy. Roughly speaking, cross-entropy penalizes the model more heavily when it assigns low probability to the correct token. If the correct token has 90% probability, the loss is small. If it has 1%, the loss is large.

The goal of training is to minimize the average cross-entropy loss across all positions in all sequences. This is equivalent to maximizing the likelihood that the model assigns to the training data. In statistical terms, we are finding the parameters that make the observed text most probable under the model.

Training proceeds in steps. At each step:

1. A batch of token sequences is fed into the model.
2. The model predicts a probability distribution for each position.
3. Cross-entropy loss is computed by comparing predictions to actual tokens.
4. Gradients are calculated, showing how each parameter contributed to the error.
5. Parameters are adjusted in the direction that reduces the loss.

This loop repeats billions of times. Each iteration processes many positions in parallel across a large batch, so training is highly efficient despite the sequential nature of language. Over time, the model's predictions improve, its loss decreases, and it becomes better at generating text that resembles the training data.

The key insight is that this process is fully self-supervised. No human labels are required. The text supervises itself: each token serves as the target for predicting what comes after it. This is why LLMs can be trained on massive corpora without expensive annotation.

From One Word at a Time to Coherent Thought

If the model only predicts one token at a time, how does it generate coherent paragraphs, multi-step explanations, or complete programs? The answer lies in the autoregressive nature of generation and the depth of what the model learns during training.

During inference, when you ask the model a question, it generates text token by token. Each new token is fed back into the model as part of the context for predicting the next one. This creates a chain:

- Input: “Explain photosynthesis.”
- Model predicts: “Photosynthesis”
- New input: “Explain photosynthesis. Photosynthesis”
- Model predicts: “ is“
- New input: “Explain photosynthesis. Photosynthesis is”
- Model predicts: “ the“
- And so on.

Each token conditions all subsequent tokens. The model does not plan the entire response ahead of time. It generates greedily, one step at a time, using everything generated so far as context. Yet the output is coherent because during training, the model learned what sequences are plausible continuations of other sequences across enormous amounts of text.

When the model sees “Explain photosynthesis. Photosynthesis is the,” it has seen millions of similar patterns during training: scientific explanations that begin with a definition, followed by details about process, inputs, outputs, and significance. The probability distribution it produces reflects all of these learned patterns. The next token is likely to be something like “process” or “way” or “mechanism,” because those are tokens that frequently follow this pattern in the training data.

This autoregressive generation has important implications. Because each token depends on all previous tokens, errors can propagate. If the model makes a factual mistake early in its response, subsequent tokens may be conditioned on that mistake, leading to further errors. This is one source of hallucination: the model is predicting plausible continuations of its own text, not checking facts against an external knowledge base.

But the same mechanism also enables remarkable capabilities. Because the model has learned patterns of reasoning from training data that contains explanations, proofs, and step-by-step derivations, it can generate reasoning-like text when prompted appropriately. It does not “reason” in the human sense, but it has internalized statistical regularities of reasoning processes and can reproduce them.

The fundamental idea, then, is this: next-token prediction is a simple objective that becomes extraordinarily demanding at scale. A model forced to minimize prediction error across trillions of tokens must learn grammar, facts, world knowledge, discourse structure, and domain conventions. These learned patterns enable it to generate coherent, knowledgeable, and sometimes insightful text through autoregressive generation, one token at a time. Everything else in this book builds on this foundation.

More Challenging Prediction Tasks

To fully appreciate the demands of next-token prediction, let us consider increasingly complex scenarios where prediction requires substantial implicit knowledge.

Long-distance dependencies. Consider: "The keys that John put in the drawer before he left for work this morning after checking the locks one last time while singing his favorite song are missing." When predicting after "are," the model must track that "keys" is the subject, despite a long intervening relative clause. The distance between subject and verb is substantial, yet grammatical agreement requires predicting a plural verb form.

Implicit causality. Consider: "Mary comforted Susan because she was upset." Who does "she" refer to? World knowledge tells us that the person being comforted is typically the one who is upset, so "she" refers to Susan. But the sentence structure alone is ambiguous. A model must learn statistical regularities of how emotions and actions co-occur to resolve this correctly.

Numerical reasoning. Consider: "I had 15 apples. I gave 4 to Alice, then bought 7 more at the store, and ate 2 on my way home. How many apples do I have?" Solving this requires tracking multiple arithmetic operations across a narrative. The model must parse numbers, apply operations in sequence, and maintain a running total, all while generating natural language. It does not have an internal calculator; it predicts tokens that happen to encode the correct arithmetic.

Code generation. Consider completing a Python function:

```
def merge_sorted_lists(a, b):
    result = []
    i, j = 0, 0
    while i < len(a) and j < len(b):
        if a[i] < b[j]:
            result.append(a[i])
            i += 1
        else:
```

The next tokens should complete the merge sort logic: `result.append(b[j])`, `j += 1`, followed by cleanup loops for remaining elements. Predicting this requires understanding algorithmic structure, variable scope, and programming conventions. The model has learned these patterns from seeing millions of code examples during training.

Style transfer. Consider the same fact expressed in different styles:

- Formal: "The precipitation levels exceeded normal thresholds throughout the quarter."
- Casual: "It was way rainier than usual this quarter."
- Technical: "Quarterly rainfall averaged 145% of the historical mean."

A model trained on diverse text learns to recognize and reproduce these stylistic variations. When prompted in a particular style, it activates the corresponding patterns. This is not explicit style labeling; it is implicit learning from distributional differences in vocabulary, syntax, and phrasing.

Cross-lingual transfer. Consider a model trained on multilingual text. If it sees "The capital of Germany is Berlin" and later encounters "La capitale de l'Allemagne est," it can predict "Berlin" even though the prompt is in French. The model has learned cross-lingual alignments: that "Allemagne" corresponds to "Germany," that both sentences express the same factual relationship, and that the answer should be the same entity. This ability emerges from seeing parallel and translated content during training, not from explicit translation rules.

Each of these examples demonstrates that next-token prediction, when applied at scale across diverse data, forces a model to learn representations that capture syntax, semantics, world knowledge, numerical reasoning, programming logic, style, and cross-lingual relationships. None of these capabilities is explicitly specified in the training objective. They all emerge as byproducts of the single goal: predict the next token accurately.

This emergent complexity is both the power and the mystery of large language models. A system optimized for a narrow statistical task develops broad, flexible capabilities that resemble understanding, reasoning, and creativity. Understanding how this happens requires examining every component of the system, from tokenization through architecture through training through inference, which is what this book does.

Chapter 2: From Text to Numbers - Tokenization and Preprocessing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Why Models Cannot Read Words

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Building a Vocabulary: Tokens vs. Words

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Byte-Pair Encoding and Subword Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Data Collection: Where the Training Corpus Comes From

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Tokenization Edge Cases and Their Consequences

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Impact of Sequence Length on Training

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 3: Embeddings - Meaning as Vectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Embedding Layer: A Lookup Table with Power

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Semantic Geometry: Why Similar Words Are Nearby

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Positional Encodings: Giving the Model a Sense of Order

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

What an Embedding Vector Actually Contains

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Embeddings in Context: From Static to Dynamic Representations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Mathematics of Cosine Similarity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Embedding Dimensionality and Capacity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 4: The Transformer Architecture - An Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Before Transformers: Why RNNs Could Not Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Big Picture: Layers, Blocks, and Data Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Feed-Forward Networks: The Hidden Computation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Residual Connections and Layer Normalization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Encoder, Decoder, and Decoder-Only Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Parameter Counts and Model Sizing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Complete Forward Pass: A Numerical Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 5: Self-Attention - The Heart of the Transformer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

What Does It Mean for One Token to Attend to Another?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Queries, Keys, and Values: The Attention Mechanism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Softmax: Turning Scores into Weights

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Multi-Head Attention: Different Perspectives on the Same Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Tensor Shapes and Dimensionality Tracking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Computational Complexity: The Quadratic Bottleneck

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Attention Patterns: What the Weights Reveal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 6: Training Mechanics - Loss, Gradients, and Descent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Measuring Failure: The Cross-Entropy Loss Function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Backpropagation: How Errors Flow Backward

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Gradient Descent: Nudging Parameters in the Right Direction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Adam and the Art of Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Tracing Gradients Through a Transformer Block

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Learning Rate Schedules: Warmup and Cosine Decay

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Gradient Clipping and Training Stability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Loss Landscape and Optimization Dynamics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 7: Pretraining - Learning from the Entire Internet

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Pretraining Loop: One Step at a Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

What the Model Learns: Patterns, Facts, and Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Scaling Laws: Why Bigger Is Better (and Why)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Emergent Abilities: When More Parameters Change Everything

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Distributed Training: Parallelism at Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Phases of Learning During Pretraining

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Data Quality Versus Quantity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 8: Knowledge Representation - What Lives in the Weights?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Parameters as Compressed Knowledge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Finding Facts Inside Neural Networks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Knowledge Neurons and Factual Localization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Superposition Hypothesis: More Features Than Dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Sparse Autoencoders: Disentangling Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Feed-Forward Networks as Key-Value Memories

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Induction Heads: The Circuit Behind In-Context Learning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Limits of Current Understanding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Geometry of Reasoning: How Chains of Thought Emerge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

What Models Do Not Know (and Why They Hallucinate)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 9: Instruction Tuning - Teaching the Model to Obey

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

From Completion Engine to Assistant

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Supervised Fine-Tuning: Learning from Examples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Reinforcement Learning from Human Feedback

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Reward Model Training: From Preferences to Scores

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

PPO Optimization: Reinforcement Learning in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Direct Preference Optimization: A Simpler Alternative

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Trade-Off Between Capability and Alignment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 10: Inference - How a Response Is Generated, Token by Token

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Inference Loop: One Token at a Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

KV Caching: Avoiding Redundant Computation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Temperature and Sampling: Controlling Creativity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Greedy Decoding, Beam Search, and Their Trade-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Context Windows and Computational Cost

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Prefill Versus Decode: Two Different Computational Regimes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

KV Cache Memory: The Hidden Cost

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Speculative Decoding: Generating Multiple Tokens Per Step

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Repetition Penalties and Generation Controls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chapter 11: Reasoning and Emergent Capabilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Did We Train Models to Reason? (No. So How?)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

In-Context Learning: Prompting as Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Chain-of-Thought: Making Reasoning Visible

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Boundary Between Pattern Matching and Reasoning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Emergence Debate: Real Phenomenon or Measurement Artifact?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Systematic Evaluation of Reasoning Capabilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

What In-Context Learning Reveals About Model Capabilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Conclusion: What We Know, What We Do Not

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

The Complete Picture: From Tokens to Intelligence-Like Behavior

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

What LLMs Achieved (and How)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

Open Questions and Honest Uncertainties

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/llmsexplained.